Was it a Dream

Relatório

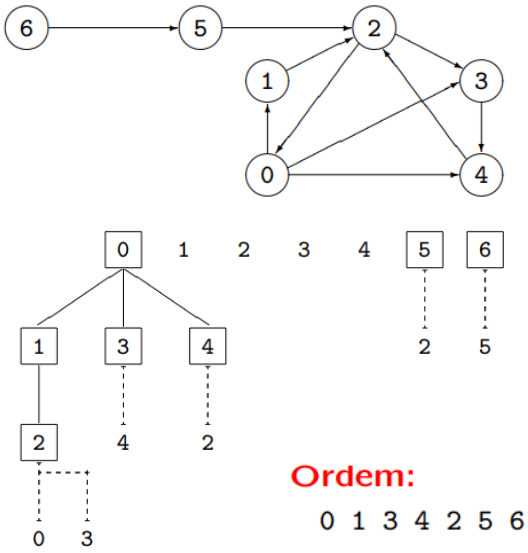
Guilherme Santana, 60182
Filipe Leão, 60191

Resolução do Problema

Para a resolução do problema apresentado neste trabalho, utilizamos o algoritmo Breadth-First Search Traversal, em português, procura em largura.

Este algoritmo percorre a estrutura de dados, neste caso uma matriz de caracteres, de forma sistemática e em ordem de largura. Começa no ponto de partida, escolhido por input, e vai explorando todos os caminhos possíveis, esquerda, direita, cima e baixo, antes de passar para a posição seguinte.

O algoritmo mantém uma fila de posições a serem exploradas. A posição inicial é colocada na fila e marcada como visitada. Em seguida enquanto a fila não estiver vazia o algoritmo remove a posição na frente da fila, explora os caminhos possíveis, marca-os como visitados e os caminhos que não terminam fora do mapa ou num no já visitado são colocados na parte de trás da fila. Este processo é repetido até a fila ficar vazia ou até chegar ao Hole.



Complexidade Temporal

Na nossa solução, para além de recorrermos à matriz de jogo, que é percorrida para ser colocada uma limitação nas suas bordas, com um X, utilizamos também uma matriz *found*, que guarda todas as posições que já foram visitadas do tamanho da matriz de jogo a ser percorrida, e uma fila de espera.

Assim, a complexidade temporal da nossa solução é:

- Percorrer a matriz de jogo - $\Theta(R \cdot C)$
- Criação e inicialização da matriz *Found* - $\Theta(R \cdot C)$
- Execução do algoritmo de resolução - $\mathrm{O}(R \cdot C)^2$
- Total - $\mathrm{O}(R \cdot C)^2$

Este resultado pode ser traduzido pelo numero possível de posições de expansão do algoritmo, sendo este certamente menor que o numero de elementos da matriz, mas provavelmente diretamente proporcional ao mesmo

Complexidade Espacial

Na nossa implementação, fazemos uso de duas matrizes cujo seu tamanho depende do tamanho de jogo. Para alem das matrizes utilizamos também uma fila de espera. Concluímos que a complexidade espacial iria ser:

- Matriz de jogo - $\Theta(R \cdot C)$
- Matriz *Found* - $\Theta(R \cdot C)$
- Fila *waiting* - $\mathrm{O}(R \cdot C)$
- Total - $\Theta(R \cdot C)$

Conclusões

Ao analisarmos o problema que nos foi apresentado neste trabalho, o primeiro algoritmo que pensamos utilizar foi o de pesquisa em profundidade. Apercebemos-mos, ao analisarmos com mais detalhe o enunciado, os problemas que esse algoritmo poderia trazer relativamente à complexidade temporal como espacial.

A solução foi otimizada por fim com o algoritmo de pesquisa em largura, visto que este oferece o melhor equilíbrio de complexidades.

Não encontramos possíveis melhorias à nossa solução.